Documentación IncliData v.0 (Claude)

1. Estructura del archivo JSON

1.1. Estructura de un Archivo JSON para Inclinómetros

El archivo tiene una estructura jerárquica con datos organizados por campañas de medición, información general y umbrales.

A continuación, se presenta un ejemplo de la estructura general de un archivo JSON para los datos de inclinómetros:

1.1.1. Estructura General

```
{
  "info": {
    // Información general del inclinómetro
  },
  "umbrales": {
    // Definición de umbrales para alertas
  },
  "YYYY-MM-DDTHH:MM:SS": {
    // Datos de una campaña específica (fecha ISO)
  },
  "YYYY-MM-DDTHH:MM:SS": {
    // Datos de otra campaña (fecha ISO)
  }
  // ... más campañas
}
```

1.1.2. Desglose de Cada Sección

a) **Bloque "info"**

Contiene información general sobre el inclinómetro, la estructura es fija:

```
"info": {
  "nom_sensor": "ID-123",          // Identificador del inclinómetro. Nombre
  "cota_1000": 500.0,              // Cota de referencia para index_0=1000
  "adquisicion": "manual",         // Método de adquisición de datos
  "disposicion": "vertical",       // Orientación del inclinómetro
  "sentido_calculo": "abajo-arriba",        // Dirección de cálculo
  "nom_campo": "Túnel Norte",              // Ubicación/proyecto
  "coordenadas": {                 // Ubicación geográfica
    "x": 123456.78,
    "y": 876543.21,
    "z": 500.0
  }
}
```

b) **Bloque "umbrales"**

Define los umbrales para evaluación de desplazamientos:

```
"umbrales": {
  "deformadas": {
    "umbral1_a":"evaluación_flanco",
    "umbral2_a", "evaluación_flanco",
    "umbral1_b", "evaluación_flanco",
          ….
  },
  "valores": [
    {
      "cota_abs": 500.0,
      "depth": 0.0,
      "umbral1_a": 2.0,
      "umbral2_a": 5.0,
      "umbral3_a": 10.0,
      "umbral1_b": 2.0,
      "umbral2_b": 5.0,
      "umbral3_b": 10.0
    },
    // ... más valores a diferentes profundidades
  ]
}
```

En las deformadas se define el nombre y la evaluación_flanco, que puede tomar los valores: flanco_positivo, flanco_negativo o sin_flanco. La evaluación en el caso de flanco_positivo comprobará si hay valores mayores que la deformada. De forma análoga para flanco_negativo. En el caso de sin_flanco se usa sólo con propósitos de visualización

---

c) Bloques de Campañas

Cada campaña está identificada por su fecha y hora en formato ISO:

```
"2023-05-15T10:30:00": {
  "campaign_info": {       // Estructura de campos fija
    "index_0": 1000,            // Punto de referencia para índices
    "importador": "RST",         // Tipo de importador usado
    "instrument_constant": 1000,   // Constante específica del instrumento, paso a mm
    "reference": true,           // Si es campaña de referencia
    "active": true,              // Si está activa
    "quarentine": false,         // Si está en cuarentena
    "alarm": "por definir"       // Estado de alarma
  },
  "info_readout": {          // Información que depende de cada fabricante, se extrae del archivo raw
    "nom_campo": "Túnel Norte",           // Nombre del campo/ubicación
    "interval": 0.5,                      // Intervalo de medición
    "probe_serial": "PS-123",             // Serial de la sonda
    "reel_serial": "RS-456",              // Serial del carrete
    "reading_units": "digits",            // Unidades de medida
    "depth_units": "m",                   // Unidades de profundidad
    "operator": "Juan Pérez",             // Operador
    "fecha_campo": "2023-05-15T10:30:00"         // Fecha y hora de la campaña
    // ... otros metadatos específicos del fabricante
```

```
},
"raw": [                          //datos sin tratar del archivo. La estructura es fija
  {
    "index": 1001,                // Índice de la medición
    "cota_abs": 500.5,            // Cota absoluta
    "depth": 0.5,                 // Profundidad
    "a0": 1524,                   // Lectura A0 (raw)
    "a180": 1498,                 // Lectura A180 (raw)
    "b0": 1655,                   // Lectura B0 (raw)
    "b180": 1623                  // Lectura B180 (raw)
  },
  // ... más puntos de profundidad
],
"calc": [                         // datos transformados en mm
  {
    "index": 1001,                 // Índice de la medición
    "cota_abs": 500.5,            // Cota absoluta
    "depth": 0.5,                 // Profundidad
    "a0": 1.524,                  // Lectura A0 (procesada en mm)
    "a180": 1.498,                 // Lectura A180 (procesada en mm)
    "b0": 1.655,                  // Lectura B0 (procesada en mm)
    "b180": 1.623,                // Lectura B180 (procesada en mm)
    "checksum_a": 3.022,          // Suma de control A (a0 + a180)
    "checksum_b": 3.278,           // Suma de control B (b0 + b180)
    "dev_a": 0.013,                // Desviación A ((a0 - a180)/2)
    "dev_b": 0.016,                // Desviación B ((b0 - b180)/2)
    "incr_dev_a": 0.0,             // Incremento de desviación A respecto a referencia
    "incr_dev_b": 0.0,            // Incremento de desviación B respecto a referencia
    "incr_dev_abs_a": 0.0,        // Incremento acumulado A
    "incr_dev_abs_b": 0.0,         // Incremento acumulado B
    "desp_a": 0.0,                // Desplazamiento A
    "desp_b": 0.0                 // Desplazamiento B
  },
  // ... más puntos de profundidad
],
"spike": {                         // Correcciones de spikes (si existen)
  "A": [
    {
      "index": 1010,
      "cota_abs": 505.0,
      "depth": 5.0,
      "A_spk": true
    }
  ],
  "B": []                      // Lista de correcciones para el eje B
},
"bias": [                     // Correcciones de bias (si existen)
  {
    "Correccion": "Bias_1_A",
    "Selec": true,
    "Prof_inf": 20.0,
    "Prof_sup": 0.0,
    "Delta": 5.2
  },
  // ... más correcciones de bias
```

```
  ]
}
```

Los bloques “spike” y “bias” recogen las correcciones realizadas en la campaña, si existen.

Nota: las profundidades vienen definidas por tres parámetros:

- “index”: índice en números enteros absolutos que por defecto toma el valor 1000 para la primera profundidad “0,5”. Aumenta a medida que aumenta la profundidad.

  El objeto es poder gestionar tubos que tengan recrecimiento de la cota de la boca.

- “cota_abs”
- “depth"

2. Transformación a valores de desplazamiento

Los cálculos transforman los valores de la sonda en **mm**, en cada importador se incluye la lógica para transformar los datos y guardarlos en **“calc”**.

Se realizan las siguientes operaciones:

1. Se selecciona la campaña a tratar y el importador correspondiente
2. Pasos del importador:
   a. Inserta la campaña en el bloque “raw”
   b. Calcula los valores “directos” y los inserta en el bloque “calc” (función valores_calc_directos):
      – a0, …., b180
      – checksum_a, checksum_b → (a0 + a180)/2
      – dev_a, dev_b → (a0 - a180)/2
   c. Calculo los valores de deformación absoluta y los inserta en “calc” (función calcular_incrementos):
      – incr_dev_a, incr_dev_b → ($dev\_a_{(campaña)} - dev\_a_{(referencia)}$)
      – incr_dev_abs_a, incr_dev_abs_b → $incr\_dev\_a_{(campaña)} + incr\_dev\_a_{(referencia)}$. En el caso de que haya habido referencias, suma el incr_dev de la referencia
      – Cálculo de envolventes y desplazamientos:
        Para cada índice:
        - Calcula la desviación absoluta (abs_dev_a, abs_dev_b) sumando todas las desviaciones desde el índice actual hasta el final → Envolvente del tubo

          $abs\_dev\_a = \Sigma\ dev\_a_{(desde\ el\ índice\ hasta\ el\ final\ del\ tubo)}$
        - Calcula los desplazamientos (desp_a, desp_b) sumando todos los incrementos desde el índice actual hasta el final → Desplazamiento acumulado

          $desp\_a = \Sigma\ incr\_dev\_a_{(desde\ el\ índice\ hasta\ el\ final\ del\ tubo)}$

Nota: el importador en los casos de ser primera campaña pone a cero todos los valores que dependen de la campaña referencia. En caso de ser una nueva referencia, copia los valores de incr_dev_abs_a y incr_dev_abs_b de la fecha anterior para acumular el desplazamiento histórico

3. Funcionamiento de importar

Flujo de Datos en el Proceso de Importación de Archivos

El proceso de importación en la aplicación de gestión de inclinómetros sigue un flujo claro y estructurado. A continuación, resumo el flujo de datos que ocurre cuando se importa un archivo nuevo:

3.1. Selección del Archivo JSON Base

- El usuario selecciona un archivo JSON existente mediante el componente import-file-dropdown
- Este archivo sirve como base para añadir nuevas campañas
- Los datos del JSON se cargan en memoria en el store tubo
- Se extraen datos básicos como información del inclinómetro, cota y el valor index_0

3.2. Selección del Importador

- El usuario selecciona el tipo de importador adecuado según el fabricante del inclinómetro (RST, Sisgeo, Soil Dux)
- Se configura el valor index_0 para referencia posicional
- Se decide si la campaña actuará como nueva referencia

3.3. Carga de Archivos de Campaña

- El usuario sube los archivos de campaña mediante el componente import-file-upload
- Los archivos se procesan en memoria (no se guardan físicamente)
- Se decodifican de base64 a texto plano
- Se separan en líneas para su procesamiento

3.4. Procesamiento de Datos

- Se aplica la función de importación específica según el fabricante:
    - import_RST para archivos RST
    - import_Sisgeo para archivos XML de Sisgeo
    - import_soil_dux para archivos de Soil Dux
- Cada importador:

1. Extrae metadatos (fecha, hora, operador, serial de instrumento, etc.)
2. Procesa las lecturas (A0, A180, B0, B180, profundidad)
3. Calcula valores derivados (checksum, desviaciones)
4. Genera una estructura de datos estandarizada

3.5. Cálculo de Incrementos

- Para cada fecha importada:
    1. Se busca su referencia apropiada mediante buscar_referencia()
    2. Se calculan los incrementos entre la campaña actual y su referencia
    3. Se aplica calcular_incrementos() para obtener desplazamientos acumulados
    4. Se actualizan valores como incr_dev_a, incr_dev_b, desp_a, desp_b

3.6. Previsualización de Resultados

- Se genera una visualización gráfica mediante importar_graficos()
- Se muestran los datos de desplazamiento en gráficos interactivos
- Se permite al usuario revisar los resultados antes de guardar

3.7. Configuración de Campañas

- El usuario puede:
    1. Modificar fecha y hora para cada campaña
    2. Marcar campañas como activas o inactivas
    3. Marcar campañas para cuarentena
    4. Seleccionar qué campañas guardar

3.8. Guardado de Datos

- Se construye una estructura final incorporando todas las modificaciones
- Se aplica insertar_camp() para añadir las campañas al archivo JSON
- Las nuevas campañas se integran con las existentes
- Se muestra confirmación al usuario

3.9. Flujo de Transformación de Datos

Archivo Original → Carga en memoria → Decodificación →

→ Extracción de lecturas → Normalización → Cálculos derivados →

→ Cálculo de incrementos → Estructura JSON final → Archivo actualizado

Durante todo este proceso, los datos pasan de estar en archivos con formatos específicos del fabricante a una estructura de datos estandarizada en memoria, para finalmente integrarse en el archivo JSON de almacenamiento, manteniendo coherencia con los datos existentes.

4. Estructura General de la Aplicación

La aplicación es un sistema web desarrollado con Dash para la gestión, visualización y análisis de datos de inclinómetros. Está organizada en una estructura modular que facilita su mantenimiento y expansión.

```
/mi_app
  ├── app.py              # Punto de entrada principal que configura la aplicación
  ├── /assets
  │   ├── styles.css      # Estilos CSS personalizados
  ├── /pages              # Módulos de páginas de la aplicación
  │   ├── configuraciones.py # Página para configuraciones futuras
  │   ├── correcciones.py   # Página para corrección de datos (spikes, bias)
  │   ├── graficar.py       # Página para visualización gráfica de datos
  │   ├── importar.py       # Página para importación de datos
  │   ├── importar_umbrales.py # Página para importación de umbrales
  │   ├── info.py           # Página de información general
  ├── /utils              # Módulos de utilidades y funciones compartidas
  │   ├── diccionarios.py   # Diccionarios de datos comunes
  │   ├── funciones_comunes.py # Funciones generales utilizadas en toda la app
  │   ├── funciones_correcciones.py # Funciones específicas para correcciones
  │   ├── funciones_graficar.py # Funciones específicas para visualización
  │   ├── funciones_graficos.py # Funciones adicionales para gráficos
  │   ├── funciones_importar.py # Funciones específicas para importación
  │   ├── rutas.py          # Definición de rutas de la aplicación
  ├── /data               # Directorio para almacenamiento de datos
  │   └── elementos.json    # Datos de ejemplo o configuración
```

5. Descripción de Módulos y Funciones

5.1. app.py

Propósito: Archivo principal que inicializa la aplicación Dash y configura las rutas.

Funciones principales:

- app = dash.Dash(...): Inicializa la aplicación

- sidebar: Define la barra lateral de navegación
- render_page_content: Callback que maneja la navegación entre páginas

5.2. utils/diccionarios.py

Propósito: Contiene diccionarios y constantes utilizados en toda la aplicación.

Contenido:

- importadores: Mapeo de nombres de importadores a sus identificadores internos
- colores_basicos: Lista de colores básicos para gráficos
- colores_ingles: Conversión de nombres de colores de español a inglés

Dónde se usa: En importar.py para la selección de importadores y en funciones de visualización para los colores.

5.3. utils/funciones_comunes.py

Propósito: Funciones de utilidad generales utilizadas en toda la aplicación.

Funciones principales:

- debug_funcion(texto): Imprime mensaje de depuración con timestamp. **Uso**: Depuración en varios módulos, especialmente en correcciones.py.
- valores_calc_directos(index, cota_abs, depth, a0, a180, b0, b180, cte): Calcula valores derivados para cada profundidad de medición. **Uso**: En funciones de importación para la normalización inicial de datos.
- buscar_referencia(data, fecha_calc): Encuentra la fecha de referencia para una fecha de cálculo dada. **Uso**: En calcular_incrementos y funciones de corrección.
- buscar_ant_referencia(data, fecha_referencia): Encuentra la fecha anterior a una referencia dada. **Uso**: En procesos de cálculo de incrementos y referencias.
- calcular_incrementos(data, fecha_calc, fecha_referencia): Calcula los incrementos de desplazamiento entre fechas. **Uso**: Crucial para el procesamiento de datos en importar.py y correcciones.py.
- get_color_for_index(index, color_scheme, total_colors): Genera colores para visualización según el esquema seleccionado. **Uso**: En todas las funciones de visualización gráfica.
- df_to_excel(df, nombre_archivo): Exporta un DataFrame a Excel. **Uso**: Función de utilidad para pruebas y exportación de datos.
- camp_independiente(fecha_seleccionada, df): Determina si una campaña es independiente. **Uso**: En los procesos de corrección para gestionar cambios en campañas.
- asignar_colores(umbrales_tubo, colores_disponibles): Asigna colores a umbrales según reglas específicas. **Uso**: En la visualización de umbrales en gráficos.

5.4. utils/funciones_correcciones.py

Propósito: Funciones específicas para la corrección de datos de inclinómetros.

Funciones principales:

- grafico_violines(df, fecha_seleccionada): Genera gráficos de violín para análisis estadístico. **Uso**: En la página de correcciones para visualizar distribuciones.
- dict_a_df(data, variables, fechas_seleccionadas): Convierte datos de diccionario a DataFrame. **Uso**: En varias funciones de corrección para facilitar análisis.
- creacion_df_bias(calc_ref, calc_corr): Crea un DataFrame para corrección de bias. **Uso**: En el proceso de corrección de bias.
- calculos_bias_1(df_bias, df_bias_table, json_spikes): Realiza cálculos de corrección de bias. **Uso**: En los callbacks de corrección de bias.
- tabla_del_json(df, fechas): Construye una tabla a partir de datos JSON. **Uso**: En la interfaz de correcciones para mostrar datos.
- std(variables, fechas_activas, data, profundidad): Calcula desviación estándar para análisis. **Uso**: En funciones de análisis estadístico de correcciones.

5.5. utils/funciones_graficos.py

Propósito: Funciones específicas para generar gráficos a partir de datos procesados.

Funciones principales:

- importar_graficos(selected_file_data, fechas_agg): Genera gráficos para visualización de datos importados. **Uso**: En la interfaz de importación para mostrar datos recién importados.

5.6. utils/funciones_importar.py

Propósito: Funciones para la importación de datos de diferentes fabricantes de inclinómetros.

Funciones principales:

- import_RST(files, index_0, cota): Importa archivos de inclinómetros RST. **Uso**: En el módulo de importación cuando se selecciona el importador RST.
- import_Sisgeo(files, index_0, cota): Importa archivos de inclinómetros Sisgeo (XML). **Uso**: En el módulo de importación cuando se selecciona el importador Sisgeo.
- import_soil_dux(files, index_0, cota): Importa archivos de inclinómetros Soil Dux. **Uso**: En el módulo de importación cuando se selecciona el importador Soil Dux.
- insertar_camp(data, fechas_agg, selected_filename, data_path): Inserta campañas en el archivo JSON. **Uso**: Al finalizar el proceso de importación para guardar los datos.
- es_fecha_isoformat(clave): Verifica si una clave tiene formato ISO de fecha. **Uso**: En funciones de procesamiento para filtrar fechas.
- default_value(data): Extrae valores por defecto de datos cargados. **Uso**: En la interfaz de importación para pre-configurar opciones.

5.7. pages/correcciones.py

Propósito: Módulo para la corrección de datos de inclinómetros, incluyendo eliminación de spikes y corrección de bias.

Componentes principales:

- layout(): Define la interfaz de usuario para la página de correcciones.
- Callbacks para:
    - Carga y visualización de datos
    - Corrección de spikes
    - Corrección de bias
    - Visualización de resultados
    - Guardado de correcciones

Funciones destacadas:

- manejar_archivo_y_guardar: Gestiona la carga y guardado de archivos
- registrar_cambios: Mantiene un registro de cambios realizados
- corr_grafico_1: Genera gráficos de visualización principal
- graficos_spike: Genera gráficos para análisis de spikes
- actualizar_spikes_table: Actualiza la tabla de spikes
- cambios_json_bias: Aplica cambios de bias al JSON
- graficos_bias: Genera gráficos para análisis de bias

5.8. pages/graficar.py

Propósito: Módulo para la visualización gráfica de datos de inclinómetros.

Componentes principales:

- layout(): Define la interfaz para visualización de gráficos.
- Callbacks para:
    - Carga y procesamiento de datos
    - Gestión de selectores de fechas y profundidades
    - Generación de diversos tipos de gráficos
    - Configuración de visualización

Funciones destacadas:

- actualizar_graficos: Genera múltiples gráficos de visualización
- actualizar_grafico_temporal: Genera gráficos de evolución temporal

- update_slider_dates: Actualiza el slider de fechas
- update_fechas_multiselect: Actualiza selector de fechas

5.9. pages/importar.py

Propósito: Módulo para la importación de datos de diferentes fabricantes de inclinómetros.

Componentes principales:

- layout(): Define la interfaz de usuario para el proceso de importación en pasos.
- Callbacks para gestionar el proceso de importación en 5 pasos:
    1. Selección de archivo JSON base
    2. Selección de importador e index_0
    3. Carga de archivos de campaña
    4. Configuración de campañas
    5. Guardado de cambios

Funciones destacadas:

- display_dropdown_input: Muestra opciones de importador
- display_upload_section: Muestra sección de carga de archivos
- execute_function_third: Procesa los archivos subidos
- update_campaign_settings: Actualiza configuraciones de campaña

5.10. pages/importar_umbrales.py

Propósito: Módulo para la importación de umbrales desde archivos Excel.

Componentes principales:

- layout(): Define la interfaz para importación de umbrales.
- Callbacks para:
    - Carga de archivo JSON base
    - Carga de Excel con umbrales
    - Integración de umbrales en JSON
    - Guardado de cambios

Funciones destacadas:

- round_numbers_in_dict: Redondea números en un diccionario
- load_json: Carga archivo JSON base
- load_excel: Carga y procesa archivo Excel con umbrales
- save_json: Guarda cambios en el archivo JSON

5.11. pages/configuraciones.py y pages/info.py

Propósito: Páginas simples para información y configuraciones futuras.

Componentes:

- layout(): Define interfaces básicas

Flujo de Trabajo de la Aplicación

1. **Importación de Datos**
    a) El usuario selecciona un archivo JSON base
    b) Selecciona el importador adecuado y parámetros de configuración
    c) Carga archivos de campaña
    d) Configura detalles de las campañas
    e) Guarda los datos procesados
2. **2. Visualización de Datos**
    a) El usuario carga un archivo JSON
    b) Selecciona fechas y profundidades a visualizar
    c) Configura parámetros de visualización
    d) Explora diferentes tipos de gráficos
3. **3. Corrección de Datos**
    a) El usuario carga un archivo JSON
    b) Revisa datos en busca de anomalías
    c) Aplica correcciones de spikes
    d) Aplica correcciones de bias
    e) Guarda los datos corregidos
4. **4. Importación de Umbrales**
    a) El usuario carga un archivo JSON base
    b) Carga un archivo Excel con datos de umbrales
    c) Vista previa de los umbrales
    d) Guarda el archivo JSON actualizado

Relaciones y Dependencias Entre Módulos

- **app.py** depende de todos los módulos de páginas
- **pages/correcciones.py** depende de:
    - funciones_comunes.py

    - funciones_correcciones.py
    - funciones_importar.py
    - diccionarios.py
- **pages/graficar.py** depende de:
    - funciones_comunes.py
    - diccionarios.py
- **pages/importar.py** depende de:
    - funciones_importar.py
    - funciones_comunes.py
    - funciones_graficos.py
    - diccionarios.py
- **pages/importar_umbrales.py** es relativamente independiente

6. 1. Título Principal

Este es el párrafo introductorio que proporciona una visión general del documento y sus objetivos.

1.1 Sección Principal 1

Este párrafo detalla la primera funcionalidad del software. Mantén los párrafos cortos y usa un interlineado de 1.5 para mejorar la legibilidad. Ahora los márgenes están reducidos para una apariencia más compacta y moderna.

1.1.1 Subsección 1.1

Descripción detallada de la funcionalidad. Utiliza listas con viñetas para enumeraciones generales y listas numeradas para pasos secuenciales.

- Elemento 1
- Elemento 2
- Elemento 3
- Paso 1
- Paso 2
- Paso 3

2. Sección Principal 2

Detalles adicionales sobre otra funcionalidad. Usa colores neutros y suaves para resaltar información importante.

2.1 Subsección 2.1

Instrucciones detalladas con ejemplos.

Asegúrate de incluir imágenes y diagramas relevantes.

3. Tabla de Contenido

Incluye una tabla de contenido al principio del documento para facilitar la navegación.